

CENTRAL CIRCULATION BOOKSTACKS

The person charging this material is responsible for its renewal or its return to the library from which it was borrowed on or before the **Latest Date** stamped below. **You may be charged a minimum fee of \$75.00 for each lost book.**

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.

TO RENEW CALL TELEPHONE CENTER, 333-8400

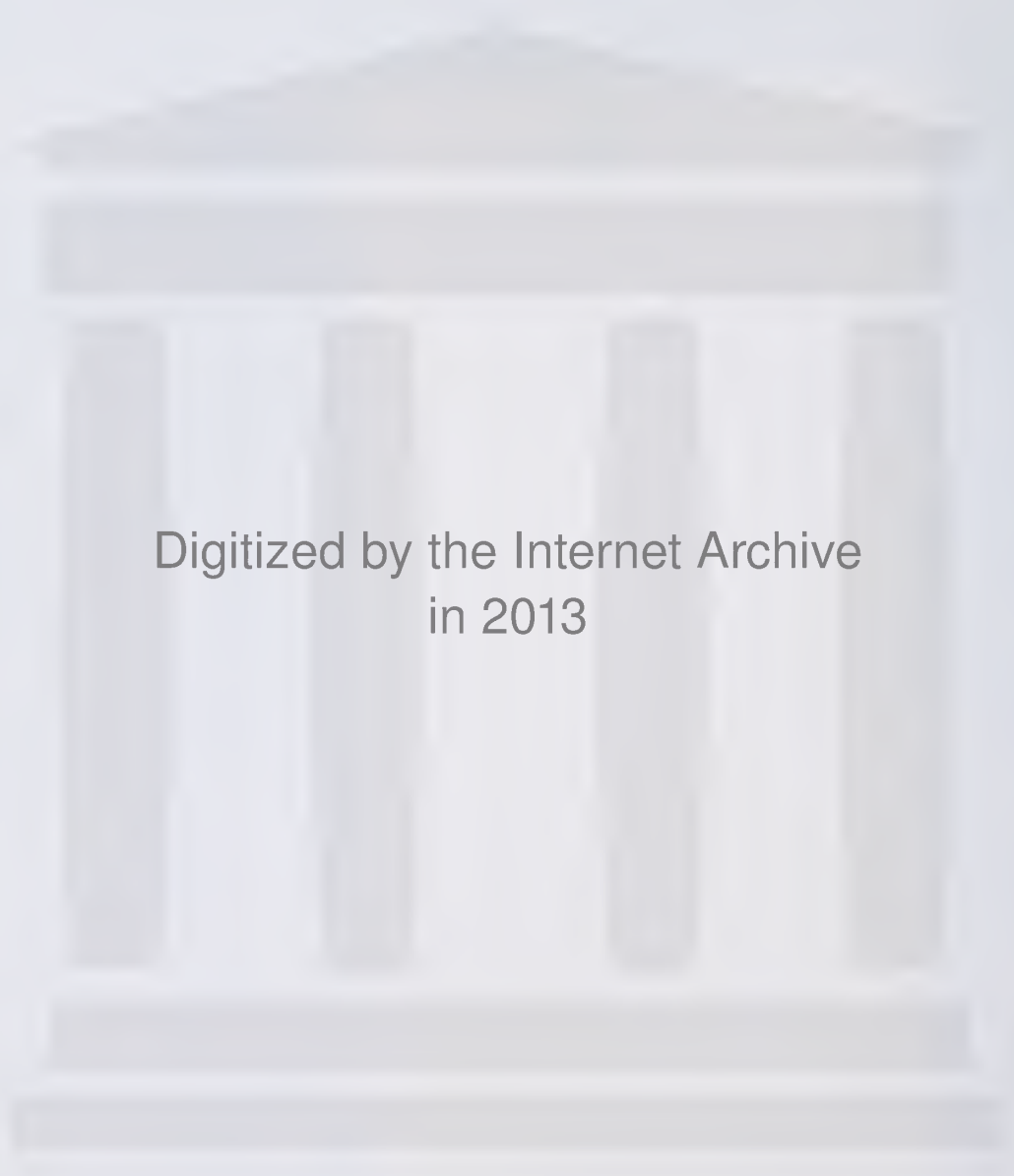
UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN

DEC 14 1998

JUL 05 2000

When renewing by phone, write new due date below
previous due date.

L162



Digitized by the Internet Archive
in 2013

<http://archive.org/details/generalizedeleme946spee>

0.001
26r
0.946
pp 2

Main

UIUCDCS-R-78-946

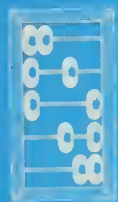
UILU-ENG 78 1738
C00-2383-0055

THE GENERALIZED ELEMENT METHOD

by

B. Speelpenning

November 1978



DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

The Library of the
JAN 10 1979
University of Illinois
at Urbana-Champaign

UIUCDCS-R-78-946

THE GENERALIZED ELEMENT METHOD

by

B. Speelpenning

November 1978

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801

PREFACE

This paper was written in 1973 to report on work that I did at the TNO-IBBC institute for Applied Technological Research in Delft, the Netherlands, and at the Delft University of Technology, also in Delft, the Netherlands. The paper was privately circulated among people at the IBM Research Center in Yorktown Heights, N.Y., and by them to others, but it was never published. Because of a number of references made to this paper in the literature, it was decided to make it available as a departmental report; no changes have been made to the original, nor any attempts to bring it up to date: it is a direct photographic reproduction of the 1973 paper.

The paper deals with the computational intricacies of assembling and solving the linear system of equations that arise from applying the finite element method to civil engineering structural analysis, but the method presented can in principle be applied wherever a network obeys Kirchhoff's laws. The "generalized element method" is an improvement over older methods of solving stiffness matrix equations. It offers a clean, very general theoretical basis, a very efficient algorithm that achieves the low operation counts of linked list techniques without any of the overhead associated with those, and great flexibility in setting up the equations. The method has been used since 1973 in the DIANA system developed at TNO-IBBC as a general purpose program for performing structural analysis and has indeed proven to be very fast, flexible and reliable. It routinely deals with systems containing 5000 equations and a "bandwidth" of up to about 400.

BY B. SPEELFENNING

0. Introduction

This paper presents a new method for solving large, sparse symmetric systems of equations derived from networks. The method is particularly suited to networks arising from using the finite element method.

The main characteristics of the method are its homogeneity and efficiency.

Homogeneity allows of automatic substructuring, interrupt and restart facilities, all completely controlled by the elimination sequence of the nodes.

For each elimination sequence, minimal operation count is achieved.

The amount of I/O can be controlled to a certain extent.

The generalized element method has evolved from a new interpretation of the wavefront method.

It may be used adaptively in combination with such methods as the hypermatrix method to optimize the use of available core.

1. Vectors and matrices: a Generalization

A vector in n -dimensional space is mostly considered as an ordered set of n real (or complex) numbers. If the vector is denoted x , then the i -th component is denoted x_i .

It is also possible to consider x as a real-valued function, defined on the set of the first n integers. The i -th component of x is then $x(i)$, the value of the function x for the argument i . The two views are equivalent, but it is the second that can be generalized immediately.

Let S be any finite set.

A vector on S is a real-valued function defined on S .

A matrix on S is a real-valued function defined on $S \times S$.

This last definition could have been less restrictive, but for the purposes of this paper it is sufficiently general.

Normal vector and matrix operations can be defined on the generalized vectors and matrices; for instance, addition of two vectors on S , multiplication of a vector on S with a scalar, or multiplication of two vectors on S forming an innerproduct. Also the dyadic product of two vectors on S may be defined, yielding a matrix on S .

The generalized vectors and matrices, however, also admit generalized operations such as the addition of two vectors on different sets:

Suppose that X_1 is a vector on S_1 , and X_2 is a vector on S_2 . The $\text{sum } X_3 = X_1 + X_2$

is defined to be a vector on $S_3 = S_1 \cup S_2$, where the value of X_3 in $a \in S_3$ satisfies

$$X_3(a) = \sum_{\substack{i=1 \\ a \in S_i}}^2 X_i(a)$$

This means that values corresponding to the same argument are added, and that zeroes are used to extend each of the vectors to S_3 .

For matrices, we define generalized addition in the same way. We will call this operation "superposition", and it is a major tool in our analysis.

This same operation is defined in normal matrix theory usually with the "help" of so-called location matrices.

We also introduce a generalized matrix-vector product, which is very useful.

Suppose A is a matrix on S and X is a vector on S' with $S \subset S'$.

The matrix-vector product $Y = A \cdot X$ is defined to be a vector on S where the value of Y in $a \in S$ satisfies

$$Y(a) = \sum_{b \in S} A(a,b)X(b)$$

This coincides with the usual formula in the case that $S = S'$.

Examples

X_1 on $\{A, B, C\}$	X_2 on $\{A, C, D\}$	X_3 on $\{A, B, C, D\}$	X_4 on $\{A, B, C, D\}$	X_5 on $\{A, B, C\}$
arg value	arg value	arg value	arg value	arg value
A a1	A b1	A a1+b1	A b1	B a2
B a2	C b2	B a2	B 0	C a3
C a3	D b3	C a3+b2	C b2	A a1
		D b3	D b3	

We note that $X_3 = X_1 + X_2$ and also $X_3 = X_1 + X_4$. It is obviously that $X_1 = X_5$;

We further note that X_4 , which contains an extra zero, is not equal to X_2 .

However, X_4 and X_2 stand in a congruence relation Q , if we define that X or S and Y on T stand in relation Q if, and only if, $X(a) = Y(a)$ for each $a \in S \cup T$ with $X(a) \neq 0$ or $Y(a) \neq 0$.

Sparse vectors, containing many zeroes, may be replaced by smaller congruent ones if considered appropriate.

2. Networks

Many systems of linear equations arise from networks.

Network elements are arbitrary sets of nodes rather than pairs of nodes as are conventional arcs in graph theory. A network element represents an explicitly known interaction between its nodes. In networks of the important class perhaps best called Kirchhoff networks, two types of variables exist, each type obeying a Kirchhoff law.

If linear behaviour of the elements is assumed, a system of equations results that relates two vectors, X and R , both on the set of all nodes.

R consists of known quantities of one type, representing the externally applied excitation; X consists of variables of the other type, representing the reaction of the network, and is to be solved.

The superposition principle inherent in Kirchhoff's law completely determines the structure of the system of equations in the way described below:

Each element e has associated with it an explicitly known element matrix E^e , which is a matrix on the set of nodes of the element. This matrix relates the unknown vector X to the "element contribution" $R^e = E^e \cdot X$.

The sum of the contributions R^e of all elements exactly equals (compensates) the externally applied excitation R .

This should be considered a direct generalization of the signal flow graphs of Mason to networks.

In what follows, element matrices will be assumed to be full or very dense.

If a physical situation is intelligently modelled into a network, this is a very natural assumption.

Applications

(a) Displacement method in structural analysis

X is the vector of nodal displacements
 R is the vector of externally applied nodal forces
 E^e are the element stiffness matrices
 R^e are the element forces acting on the nodes.

Large structures involve many thousands of nodal displacements with strong interactions. Maxwell's law guarantees that all element matrices are symmetric: $E^e(a,b) = E^e(b,a)$.

If constraints are taken into account, the system is positive-definite, roundoff being no serious problem on modern, large word computers.

A method of handling constraints compatible with our input format (network model) is described in (7).

(b) Electrical network theory

X is the vector of nodal potentials
 R is the vector of externally applied nodal currents
 E^e are the element admittance matrices
 R^e are the currents through the wires connecting the element with its nodes.

As a rule, electrical networks are smaller than networks arising from the use of the displacement method. Complex numbers and lack of symmetry (solid state devices) still may make storage considerations important.

A reference node is required to define a unique solution.

From now on, it is assumed that all element matrices will be symmetric as in the displacement method. However, many remarks remain valid without that assumption.

3. Element pool

The stated equilibrium condition $\sum_{e \in e} R^e = R$ can be converted into $\sum_{e \in e} (E^e \cdot X) = R$. Under our definition, distributive law still holds; application leads to

$$(\sum_{e \in e} E^e) \cdot X = R, \text{ or } S \cdot X = R, \text{ in which}$$

$S = \sum_{e \in e} E^e$ is the overall network matrix describing the total set of equations.

Several writers note that S is very sparse, and many attempts have been made to exploit this sparsity during the decomposition of S . For the actual decomposition, most writers start from the fully assembled network matrix, stored in some convenient way (1,3,5).

This very sparsity of the network matrix, however, indicates that a matrix is not the right way to communicate the network equations to a computer.

The most direct way to communicate them is by a file containing representations of the element matrices, including the sets on which they are defined.

This file is called the element chain or the element pool, depending on whether it is organized for serial access or random access.

4. Frontal solution

The principle that superposition of some elements may wait until elimination of already completed rows of the network matrix is performed, is noted and exploited by Irons in (2).

A proof is suggested in Fig. 5.

This principle is extremely important, since it allows us to consider elimination of nodes and superposition of element matrices as dual operations related by the sole demand that elimination must take place on completed rows. Solution methods using Irons' principle are called frontal solution methods.

Optimal ordering and wavefront

The program of Irons (2) is essentially the wavefront method of Malosh and Bamford (1), converted into a frontal solution method. Optimal orderings for minimal operation count (6), or fill in (4), are not compatible with the ordering optimal for the wavefront method. This indicates that the wavefront method does not achieve minimal operation count, even if presented an ordering for minimal operation count. Analysis of this deficiency gave rise to the generalized element method, which can be characterized as a wavefront-like frontal solution method, fully compatible with an ordering for minimal operation count.

5. Generalized elements

With each element is associated a set of nodes, the "element set" and an element matrix. We use the term element to cover both element set and element matrix.

(a) If two element matrices are superposed, a matrix results, which is defined on the union U of the element sets involved. U and the resulting matrix together may be regarded as an element; the matrix represents the now explicitly known interaction between the nodes of U , this interaction being caused by the joint influence of the two elements. These elements may be replaced in the element pool by the new, generalized element.

Superposition thus results in the decrease by one of the number of elements in the element pool.

(b) If a node of an element is eliminated from an element matrix, which is admissible if the element pool does not any more contain elements having the same node, we get two results. First, a row is formed, which can be used in the forward and backward substitution; the coefficients of the row almost explicitly express the eliminated node in terms of the remaining nodes.

It is well known that the processing of the involved terms of the right-hand side may be deferred till later, so we only consider the matrix. This first row is transported to a file, which is commonly called "table of factors". This file serves as an operational form of the decomposed matrix.

Second, a matrix is formed, which is defined on the set of the remaining nodes V of the element. V and the matrix together again may be regarded as an element; the terms of the matrix show explicitly the interaction between the nodes of V , this being the original interaction modified by the indirect interaction through the eliminated node. The original element may be replaced in the element pool by the new, generalized element.

Elimination thus results in the decrease by one of the number of nodes present in the element pool.

6. Generalized element method

The generalized element method gradually transforms the element pool by repeated application of superposition and elimination. Actual calculations take place in central memory, treated as an extension of the element pool residing on disk. At each moment, central memory contains an element. This element has the same relation to the elements on disk, as an accumulator has to central memory, and as a magnifying-glass has to a series of objects. Central memory must be large enough to accommodate for the generalized elements, as they are created by the process.

Elimination and superposition are controlled by the elimination sequence of the nodes according to the following algorithm:

- 1) Determine the first node to be eliminated; call it node n .
- 2) Put an empty element in core.
- 3) In the element pool, look for elements containing node n . Such an element is to be processed: read and superpose it on the element in core and delete it from the element pool.
- 4) Repeat step 3) till no such elements can be found anymore.
- 5) Eliminate node n from the element in core.
- 6) If no more nodes are to be eliminated, perform step 5).
Else, consider the next node in the elimination sequence; call it node n .
- 7) If the element now in core contains node n , return to step 3). If not, continue with step 8).
- 8) Transfer the element in core to the element pool on disk (except when the element in core happens to be empty). If no more nodes are to be eliminated, stop. Else, return to step 2).

Two remarks are in order here.

- (a) It is the possibility of performing step 8), the "cleaning up", that distinguishes this method from wavefront and causes it to achieve the minimally possible operation count for the given elimination sequence. Actually, when step 7) is changed into: 7') Return to step 3), we will have the wavefront method back.
- (b) The "accumulator-analogy" noted before, can be expanded to include the important procedures used in the algorithm. We take procedures to be analogous to machine code instructions.

Compare:

procedures

put empty element in core
superpose element on core
eliminate a node from the
element in core
transfer element in core
to element pool

instructions

CIA , clear accumulator
ADD , add to accumulator
SRA , shift right accumulator
STA , store accumulator

This analogy shows that step 8) of the algorithm is a very basic one, and not just an "ad hoc" extension of wavefront.

7. Implementation of the generalized element method

i) representation of nodes

Each node is assumed to have a unique name. The function of the name of the node is to identify the node. The function of the number of a node is to establish its place in the elimination sequence of the nodes. It should be noted that these two functions are completely distinct.

We assume the names of the nodes to be "ONE", "TWO", "THREE", etc. The representation of a node will be the number corresponding to the name, i.e. "3" for the node named "THREE". Again, the number corresponding to the name of the node and the number of the node are not in any way related.

ii) Representation of an element in the element pool

An element matrix, which is a function on pairs of nodes, may be represented by a table in various ways, e.g.:

SIX TWO NINE				TWO NINE SIX				
SIX	e ₁₁	e ₁₂	e ₁₃	represents the same element matrix as:	TWO	e ₂₂	e ₂₃	e ₁₂
TWO	e ₁₂	e ₂₂	e ₂₃		NINE	e ₂₃	e ₃₃	e ₁₃
NINE	e ₁₃	e ₂₃	e ₃₃		SIX	e ₁₂	e ₁₃	e ₁₁

a)

b)

To take advantage of symmetry, it is assumed that row and column order of a table are compatible, thus excluding for instance:

SIX TWO NINE			
TWO	a_{12}	a_{22}	a_{23}
NINE	a_{13}	a_{23}	a_{33}
SIX	a_{11}	a_{12}	a_{13}

Whether Table (a) or Table (b) would be chosen to represent the element matrix, turns out to be completely immaterial.

The next step consists in representing the table by two arrays.

The first array contains the numbers in the upper half of the chosen table, stored columnwise. For our example, we choose Table (b) and form:

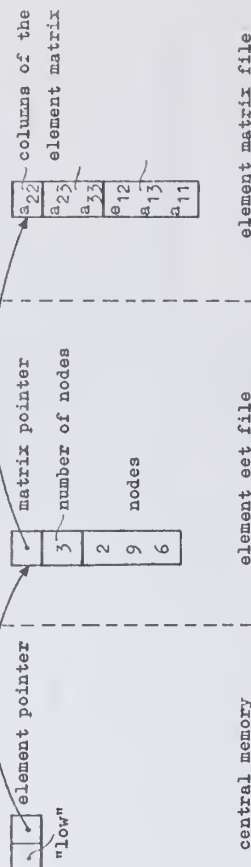
a ₂₂	a ₂₃	a ₃₃	a ₁₂	a ₁₃	a ₁₁
-----------------	-----------------	-----------------	-----------------	-----------------	-----------------

The second array contains the representations of the nodes of the element in the order implied by the table used. Continuing our example:

2	9	6
---	---	---

These two arrays suffice to represent the element.

The complete organization, including some pointer information, is shown below:



The disk organization should allow the element matrix to be retrieved one column at a time.

The set of element pointers in core and the element set file represent the topological structure of the network and might be used as input for an optimal ordering algorithm. Anyhow, the elimination sequence of the nodes is considered as input for the generalized element method.

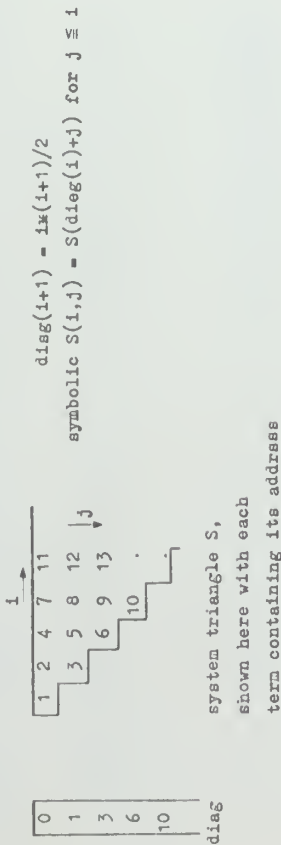
To avoid exhaustive search, in the element set file, for elements containing some specific node, an initialization step is performed.

For each element, its node first in the elimination sequence is stored in "low" in central memory. Element information in core may be sorted according to the numbers of the nodes in "low", but this is not necessary.

Of course, newly created elements that are to be stored on disk must be included in the same organization.
An element can be deleted from the pool by merely changing the node in "low" into a non-existing one, or changing its number into infinite.

iii) representation of an element in core

The representation of an element in core follows the same lines as the representation of an element on disk, but some additional information is maintained according to the magnifying function of core noted before.
The element in core is stored in the "system triangle".
The system triangle is an array in open ended form. If the element in core has n nodes, the first $n \times (n+1)/2$ terms of the system triangle contain the upper half of the element matrix; the rest contains zeroes.
At the beginning the index array "diag" is formed; it contains the addresses in the system triangle of the diagonal terms. This array serves for addressing the system triangle and does not change at all during the decomposition process.



As in the wavefront method, the nodes of the element currently in core are called frontal nodes. The number of frontal nodes is the current bandwidth, denoted "iband". The frontal nodes are stored in the array "front", which must accommodate for this maximum bandwidth.
Frontal node $n\text{-front}(i)$ corresponds to the i -th column of the system triangle.

To avoid looking in the array "front" in order to find a specific node, we introduce the characteristic vector, "kvec". The length of this array is the total number of nodes and the contents are defined by:

$$\begin{aligned} \text{kvec}(n) &= 1, & \text{if } n = \text{n-front}(i) \\ \text{kvec}(n) &= 0, & \text{if } n \text{ is not a frontal node} \end{aligned}$$

This array permits very fast inspection of the front, is easily updated during superposition and elimination, and makes it easy to decide whether step 8) of the algorithm has to be performed.

iv) representation of table of factors

Table of factors (6) is an operational form of the decomposed matrix. This table should suffice to find the actual solutions by forward and backward substitution.
The table of factors is formed one row at a time. Each node that is eliminated, yields one row; moreover, each time that step 8) is performed, a row containing the current frontal nodes is added to the table.
The decomposed rows have the following format:



"ind" is the number of the column of the triangle that contained the eliminated node; the u, u' and d correspond to the u and d of the appendix.

The rows resulting from the execution of step 8) have the format depicted in the diagram:



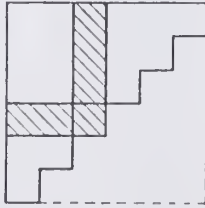
v) elimination of a node

A graphical description of Gaussian elimination is given in Appendix II. This algorithm is applied on the system triangle, taking into account the specific storage scheme used. The pivotal column (b) contains no structural zeroes, because of the nature of the generalized element method. (Changing terms in the original element matrices would destroy any zero, not called "structural".)
This means that subtraction of the dyadic product $b^{-1}b'$ changes all the terms of the remaining matrix. It can be seen now that elimination of a node will take $\text{iband} \times (\text{iband} + 1)/2$ operations.

As the elimination is the most critical part of the whole algorithm, it is very important to program the elimination algorithm as fast as possible.
In fact, many of our former representations have been chosen to permit a more efficient elimination routine.

The Gaussian elimination algorithm of Appendix II is modified in three ways. All modifications arise from the representation chosen for the element in cors .

- (a) Because of symmetry, only half of the matrix is stored. This implies that rows and columns of the square, symmetric matrix, show themselves in the system triangle as a mixture of a row and a column, called "system column". For example, the third system column with $\text{iband}=6$:



- (b) In general, the node to be eliminated is not associated with the first system column. This causes no serious problems, but it tends to make the routine more difficult to grasp.

- (c) The algorithm is to leave the remaining element matrix in the same required representation. That is, the remaining element matrix is to have its terms consecutively followed by a range of zeroes. This means that, besides a subtraction, a move is to be performed.

We avoid an extensive explanation of the Gaussian algorithm and present instead a complete Fortran subroutine, called "Gauss".

"S" is the name of the system triangles, "UBUF" is a buffer being filled with the decomposed row.

In labelled common, the current bandwidth "IBAND" is input; also "IND", giving the system column number containing the pivotal term, and "IPIV", the index in S of the pivotal term itself. (actually, $\text{IPIV}=\text{diag}(\text{IND}+1)$)

```
SUBROUTINE GAUSS(S,UBUF)
  DIMENSION S(1),UBUF(1)
  COMMON/GENEL4/IBAND,IND,IPIV
  LOGICAL SKIP1,SKIP2
  SKIP1 = IND.EQ. 1
  SKIP2 = IND.EQ. IBAND
  IND1 = IND - 1
  IND2 = IND + 1
  KGAP = IND
```

```
  D = 1./S(IPIV)
  INFL = IPIV - IND1
  UBUF(IND) = D
  KPUT = 0
  IF(SKIP1) GOTO 20
  DO 10 M = 1,IND1
    B = S(INFL)
    INFL = INFL + 1
    UBUF(M) = B * D
    DO 10 N = 1,M
      KPUT = KPUT + 1
      S(KPUT) = S(KPUT) - B * UBUF(N)
    10 IF(SKIP2) GOTO 60
    DO 50 M = IND2,IBAND
      INFL = INFL + KGAP
      B = S(INFL)
      UBUF(M) = B * D
      IF(SKIP1) GOTO 40
      DO 30 N = 1,IND1
        KPUT = KPUT + 1
        KGET = KPUT + KGAP
        S(KPUT) = S(KGET) - B * UBUF(N)
      30 KGAP = KGAP + 1
      DO 50 N = IND2,M
        KPUT = KPUT + 1
        KGET = KPUT + KGAP
        S(KPUT) = S(KGET) - B * UBUF(N)
      50 DO 70 K = 1,KGAP
        KPUT = KPUT + 1
        S(KPUT) = 0.
      70 IBAND = IBAND - 1
      RETURN
    END
```

We remark several things about the subroutine:

- a. The operation performed in the innerloops has the format

$$S(KPUT) = S(KGET) - B * UBUF(N)$$

As B and UBUF(N) will not be zero in the generalized element method, the only loss of efficiency that can occur is in subtracting a non-zero number from zero.

We neglect this loss completely, when we claim that the generalized element method achieves minimal operation count.

b. This Fortran routine appears to be very efficient. We remarked, however, that if this Fortran routine is compiled for the CDC 6600 computer, with the FTX-compiler, all possibilities of the CDC 6600 pipeline processor are still not used.

By coding the routine directly into COMPASS, the assembler language of CDC 6600, we realized profits of 30% depending on the bandwidth.

In this kind of processors, a multiplication count is not at all sufficient to estimate the running time. Access time of core very often becomes the limiting factor, if parallelism is optimized.

Exactly because of this, the fastest routines are those in which address calculation is incremental, no or hardly any indirect addressing is performed, and in which, above all, no explicit zero testing of real numbers is done.

The generalized element method is very well adapted to pipeline processors, because it treats the decomposition of a sparse matrix as a series of eliminations of nodes from full element matrices.

c. The output "UBUF" is to be extended with "IBAND" and "IND" to constitute a row of the table of factors.

d. The logic required for updating the front information in "FRONT" and "KARVEC" is not included in the given subroutine. In this paper, we will only give an example:

before elimination;

```
IBAND = 7      FRONT: 7 10 11 12 6 5 16
KARVEC: 0 0 0 0 6 5 1 0 0 2 3 4 0 0 0 7 0 0
(index: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18)
```

Suppose that node ELEVEN has the lowest number and is to be eliminated.

We find $IND = KARVEC(11) = 3$ and $IPIV = 3 \times (3+1)/2 = 6$.

The parameters for the Gauss-routine have been found; we now display the front as it has to be after elimination:

```
IBAND = 6      FRONT: 7 10 12 6 5 16
KARVEC: 0 0 0 0 5 4 1 0 0 2 0 3 0 0 0 6 0 0
(index: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18)
```

vi) superposition of an element

The superposition procedure consists of two steps.

e) First, the element set is read and for each node of the set an appropriate system column has to be found. If the node was already in the front, the

corresponding column of the system triangle can be found in KARVEC; if the node was not in the front, it becomes a frontal node, a new system column is reserved for this node, the FRONT and KARVEC arrays are updated, and IBAND is increased by one.

An example:

Suppose IBAND = 7 and the frontal nodes are determined by

```
FRONT: 6 5 9 12 10 4 16
KARVEC: 0 0 0 6 2 1 0 0 3 5 0 4 0 0 0 7 0 0
(index: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18)
```

Suppose furthermore that an element is read with element set {12, 5, 17, 8, Note that nodes TWELVE, FIVE and SIX are already in the front, but SEVENTEEN and EIGHT are not.

On completion of phase (e) we have

```
IBAND = 9
FRONT: 6 5 9 12 10 4 16 17 8
KARVEC: 0 0 0 6 2 1 0 9 3 5 0 4 0 0 0 7 8 0
(index: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18)
```

and KOLS: 4 2 8 9 1

in which KOLS denotes the system columns associated with the nodes of the element.

It is in this phase that overflow of core can easily be predicted.

For instance, if 100K words are available, IBAND may not exceed about 400. (Program space and additional arrays taken into account.)

If overflow occurs, step 8) of the algorithm is performed and the process is halted. The element pool still represents a network, and can be input in any other equation solving routine that does not have this limitation. The hypermatrix method (5) seems very promising. It probably needs experimental verification whether it is useful to couple these two methods into one program, in which overflow ceases switching to the hypermatrix method.

b) Second, the columns of the element matrix are read, one at a time.

The j-th term of the i-th column of the element matrix is added to term $jj = kols(j)$ of the system column $ii = kols(i)$ of the system triangle. In case $jj > ii$, ii and jj have to be interchanged first.

We now finish the description of the implementation of the generalized element method by noting that the forward and backward substitution can be implemented according to the same principles, and that the amount of I/O can be greatly reduced by means of buffering techniques.

The performance of the generalized element method

In (4), an extensive treatment of elimination of nodes from graphs is given; we do not repeat it here.

The method described in this paper gives the actual algorithm to implement it. A similar purpose had the program of Jensen and Parks (3), but we believe that our method compares favourably, by virtue of the storage scheme we used and the relative ease in extending the method.

In the introduction, interrupting and restarting facilities are mentioned; these can be realized by forcing the method to execute step (8) and halt.

In fact, there is no difference at all between start and restart in our method. The method accepts as input the number of nodes to be eliminated, and after elimination is finished, a new element pool results. Whenever the element pool is empty, we may perform the backward substitution (assuming that forward substitution is included in the decomposition process), regardless of the number of intermediate halts.

That the method makes it possible to get automatic substructuring, is now seen to be trivial.

If the node sequence is very bad, much time may be wasted by swapping elements in and out of core in performing repeatedly step 8). Of course, if step 8) does not result in savings in the future bandwidth, it takes just a lot of I/O. Therefore, a routine is developed, which is essentially a topological sorting, that reorders the nodes to minimize the number of times step 8) is to be performed, without affecting the operation count.

Conclusion

The generalized element method seems to be a very useful method for solving large sparse linear systems.

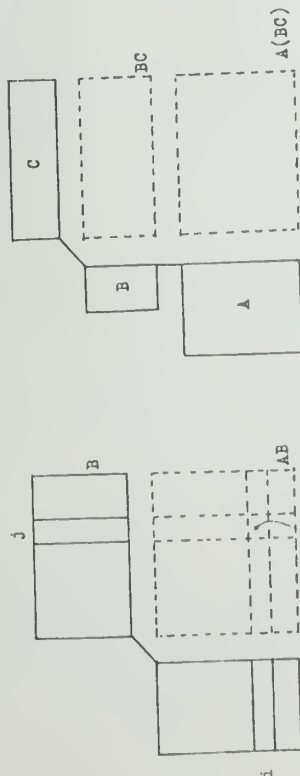
The method is general, efficient and homogeneous, and some very important facilities (substructuring, interrupting) become very simply available.

The method should also provide more insight in the nature of decomposition of networks.

APPENDIX I

References

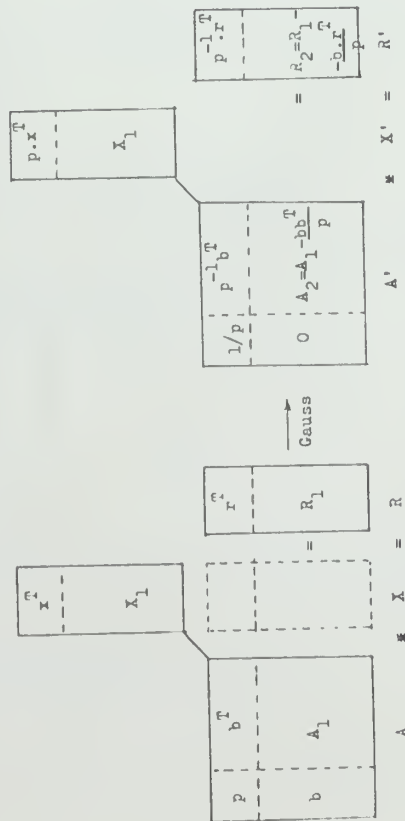
- 1) Melosh, R.J. and Bamford, R.M.
"Efficient solution of load-deflection equations"
Journal of the Structural Division, ASCE, vol. 95, Apr. 1969
- 2) Irons, B.M.
"A frontal solution program for finite element analysis"
Intern. J. Numerical Methods in Engineering, Jan. 1970
- 3) Jensen, H.G. and Parks, G.A.
"Efficient solutions for linear matrix equations"
J. Struc. Div., ASCE, vol. 96, Jan. 1970
- 4) Ogbuobiri, E.O., Tinney, W.F. and Walker, J.W.
"Sparsity-Directed Decomposition for Gaussian Elimination on Matrices"
IEEE trans. Power Apparatus and Systems, vol. PAS-89, no. 1, Jan. 1970
- 5) Fuchs, G. von, Roy, J.R. and Schrem, E.
"Hypermatrix solution of large sets of symmetric positive-definite linear equations"
Comp. Meth. in Appl. Mechanics and Engineering 1 (1972)
- 6) Tinney, W.F. and Walker, J.W.
"Direct solutions of sparse network equations by optimally ordered triangular factorisation"
Proc. IEEE, vol. 55, Nov. 1967
- 7) Tolman, F.P. and Spealpenning, B.
"The element-chain concept in finite element based programming"
Internal report, TNO Institute for Applied Scientific Research, BI-72-85, Dec. 1972
- 8) Spealpenning, B.
"The generalized element method", Preliminary Report
Notices A.M.S., 73T-C18, Feb. 1973



a) graphical notation for matrix-multiplication AB

b) graphical notation for $A(BC)$

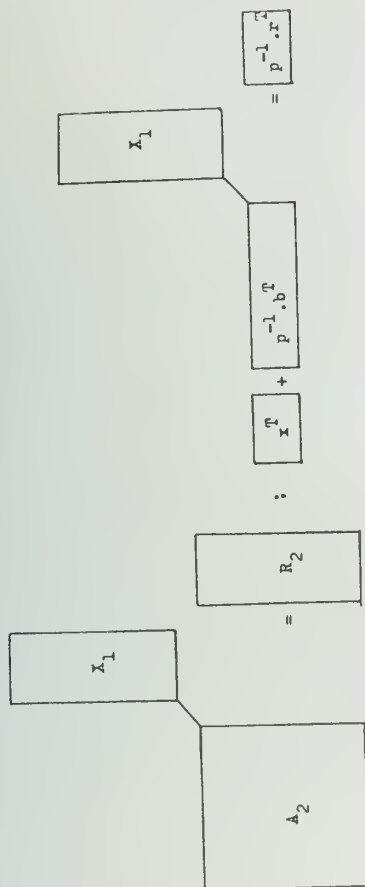
the dotted lines are not part of the notation



a) matrix equation, A and A_1 symmetric

b) transformed equation, first unknown eliminated

Figure 2. Gaussian elimination

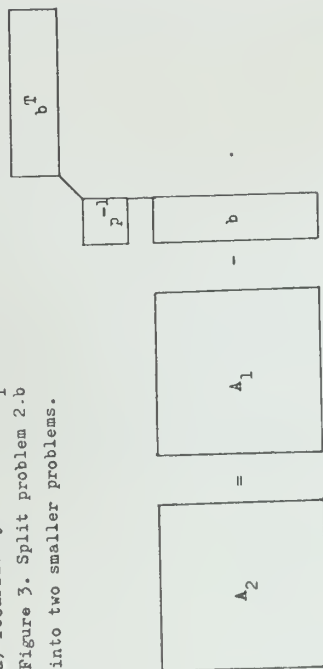


a) recursively solve for X_1

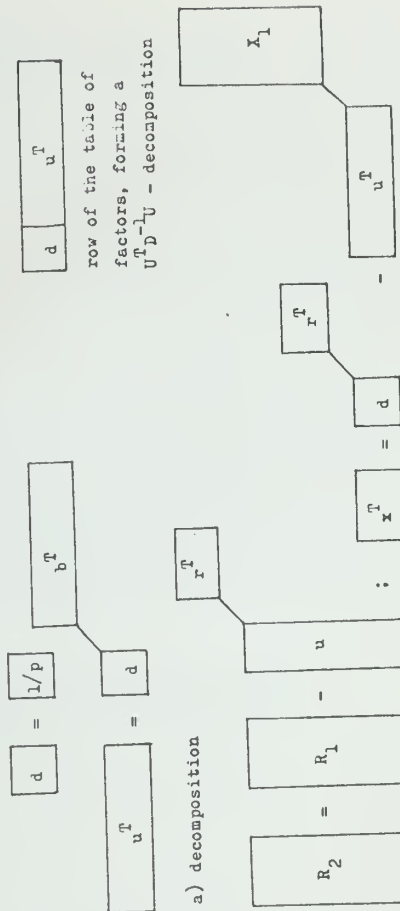
b) with X_1 found, solve for x^T

x^T

Figure 3. Split problem 2.b into two smaller problems.



a) decomposition



b) forward substitution

c) backward substitution

Figure 4. Three stages of the computation

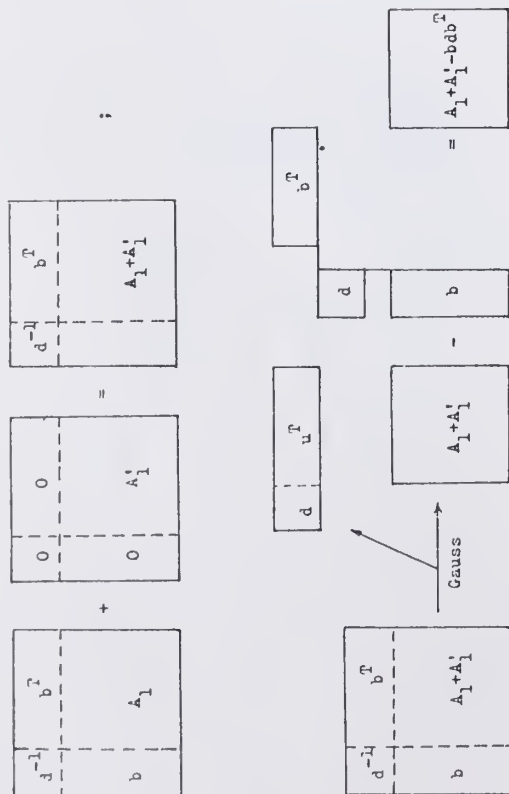
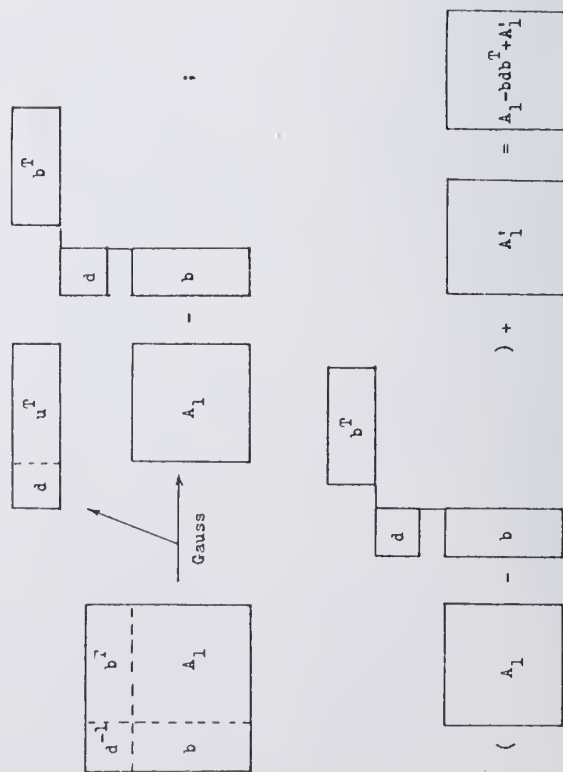


Figure 5. a) first superposition, then elimination



b) first elimination, then superposition

U. S. ATOMIC ENERGY COMMISSION
UNIVERSITY-TYPE CONTRACTOR'S RECOMMENDATION FOR
DISPOSITION OF SCIENTIFIC AND TECHNICAL DOCUMENT

(See Instructions on Reverse Side)

1. AEC REPORT NO.

C00-2383-0055

2. TITLE

THE GENERALIZED ELEMENT METHOD

3. TYPE OF DOCUMENT (Check one):

- ☒ a. Scientific and technical report
☐ b. Conference paper not to be published in a journal:

Title of conference _____

Date of conference _____

Exact location of conference _____

Sponsoring organization _____

- ☐ c. Other (Specify) _____

4. RECOMMENDED ANNOUNCEMENT AND DISTRIBUTION (Check one):

- ☒ a. AEC's normal announcement and distribution procedures may be followed.
☐ b. Make available only within AEC and to AEC contractors and other U.S. Government agencies and their contractors.
☐ c. Make no announcement or distribution.

5. REASON FOR RECOMMENDED RESTRICTIONS:

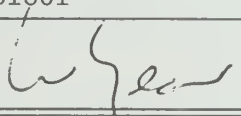
6. SUBMITTED BY: NAME AND POSITION (Please print or type)

C. W. Gear
Professor and Principal Investigator

Organization

Department of Computer Science
University of Illinois U-C
Urbana, Illinois 61801

Signature



Date

November 1978

FOR AEC USE ONLY

7. AEC CONTRACT ADMINISTRATOR'S COMMENTS, IF ANY, ON ABOVE ANNOUNCEMENT AND DISTRIBUTION RECOMMENDATION:

8. PATENT CLEARANCE:

- ☐ a. AEC patent clearance has been granted by responsible AEC patent group.
☐ b. Report has been sent to responsible AEC patent group for clearance.
☐ c. Patent clearance not required.

JUN 1 1979

BIBLIOGRAPHIC DATA SHEET		1. Report No. UIUCDCS-R-78-946	2.	3. Recipient's Accession No.
4. Title and Subtitle THE GENERALIZED ELEMENT METHOD				5. Report Date November 1978
				6.
7. Author(s) B. Speelpenning				8. Performing Organization Rept. No. R-78-946
9. Performing Organization Name and Address Department of Computer Science University of Illinois U-C Urbana, Illinois 61801				10. Project/Task/Work Unit No.
				11. Contract/Grant No. US ENERGY/ EY-76-S-02-2383
12. Sponsoring Organization Name and Address U. S. Department of Energy Chicago Operations Office 9800 South Cass Avenue Argonne, Illinois 60439				13. Type of Report & Period Covered research
				14.
15. Supplementary Notes				
16. Abstracts This paper presents a new method for solving large, sparse symmetric systems of equations derived from networks. The method is particularly suited to networks arising from using the finite element method. The main characteristics of the method are its homogeneity and efficiency. Homogeneity allows of automatic substructuring, interrupt and restart facilities, all completely controlled by the elimination sequence of the nodes. For each elimination sequence, minimal operation count is achieved. The amount of I/O can be controlled to a certain extent. The generalized element method has evolved from a new interpretation of the wavefront method. It may be used adaptively in combination with such methods as the hypermatrix method to optimize the use of available core.				
17. Key Words and Document Analysis. 17a. Descriptors finite element method, sparse linear equations, superelements, computational linear algebra, wavefront method, DIANA system, structural analysis, stiffness matrix.				
17b. Identifiers/Open-Ended Terms				
17c. COSATI Field/Group				
18. Availability Statement unlimited		19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 11	
		20. Security Class (This Page) UNCLASSIFIED	22. Price	



UNIVERSITY OF ILLINOIS-URBANA



3 0112 039311334